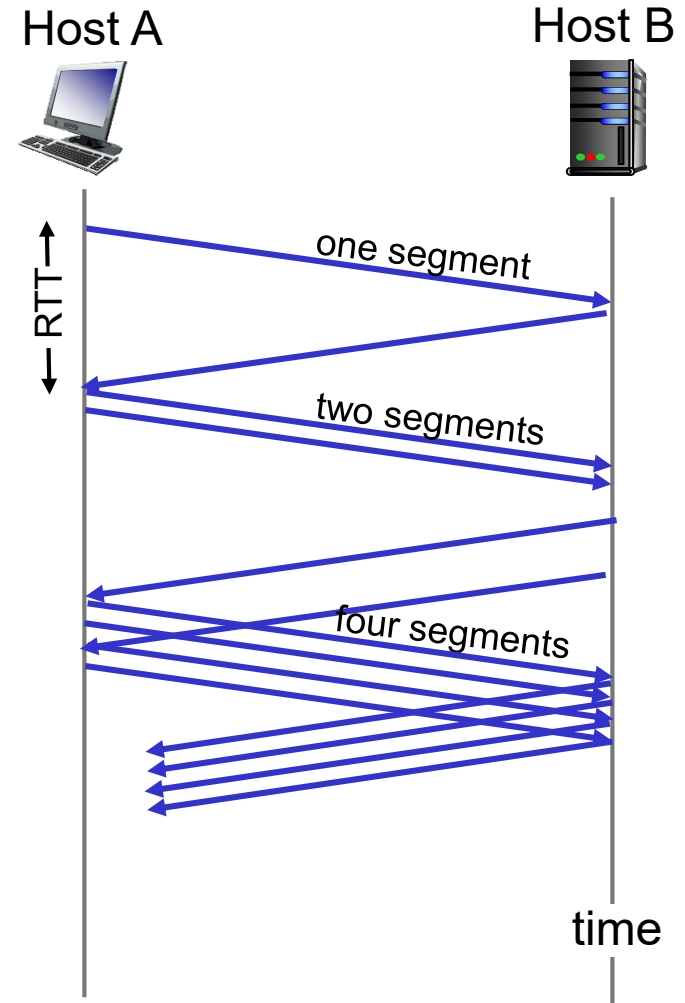# ECE 440 - Introduction to Computer Networks

Spring 2020

TCP Slow Start

- **when connection begins, increase rate exponentially until first loss event:**
  - initially `cwnd` = 1 MSS
  - double `cwnd` every RTT
  - done by incrementing `cwnd` for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast

Host A

Host B

RTT

one segment

two segments

four segments

time

TCP: detecting, reacting to loss

- loss indicated by timeout:
  - `cwnd` set to 1 MSS;
  - window then grows exponentially (as in slow start) to threshold, then grows linearly
- loss indicated by 3 duplicate ACKs: TCP RENO
  - dup ACKs indicate network capable of delivering some segments
  - `cwnd` is cut in half window then grows linearly
- TCP Tahoe always sets `cwnd` to 1 (timeout or 3 duplicate acks)
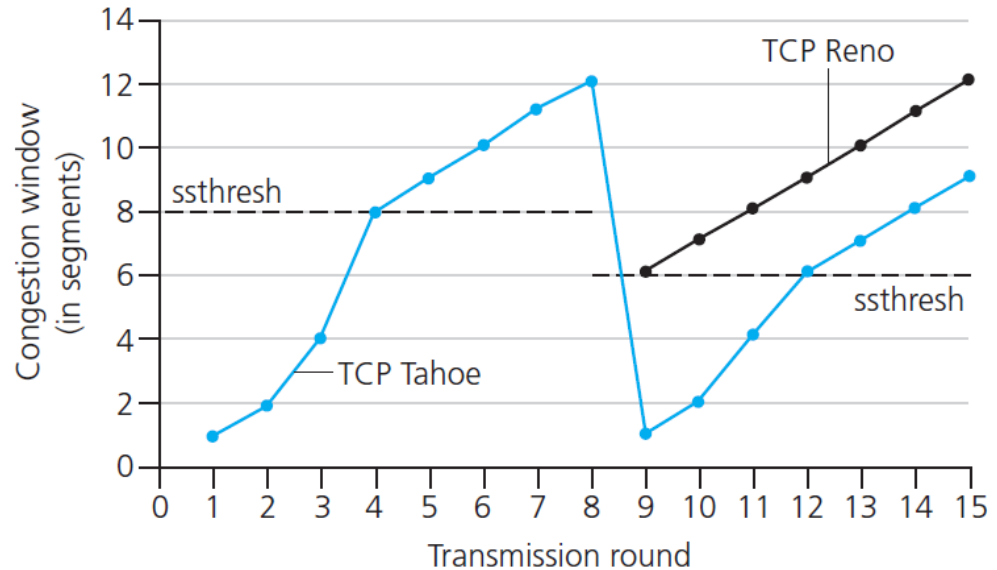
TCP: switching from slow start to CA

Q: when should the exponential increase switch to linear?

A: when **cwnd** gets to 1/2 of its value before timeout.

Implementation:

- variable **ssthresh**
- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event

TCP throughput

- avg. TCP thruput as function of window size, RTT?
  - ignore slow start, assume always data to send
- W: window size (measured in bytes) where loss occurs
  - avg. window size (# in-flight bytes) is ¾ W
  - avg. thruput is 3/4W per RTT

$$\text{avg TCP thruput} = \frac{3}{4} \frac{W}{RTT} \text{ bytes/sec}$$
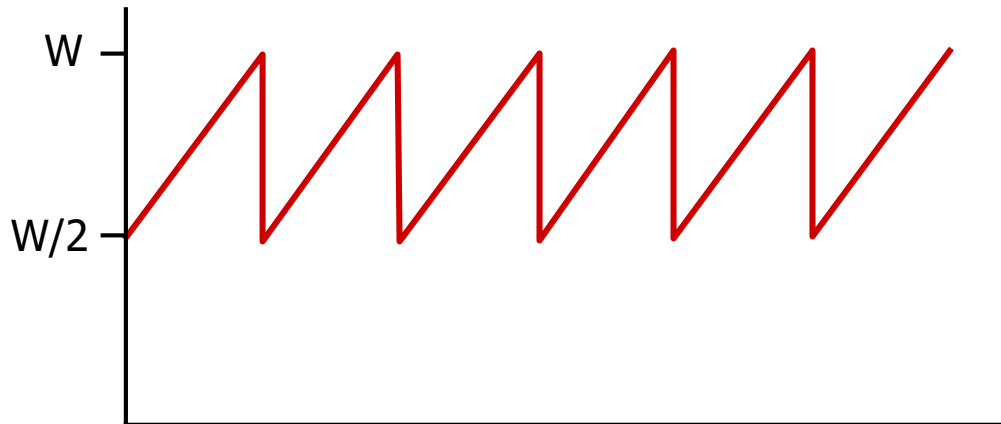
TCP Futures: TCP over "long, fat pipes"

- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires W = 83,333 in-flight segments
- throughput in terms of segment loss probability, L [Mathis 1997]:

$$\text{TCP throughput} = \frac{1.22 \cdot MSS}{RTT \sqrt{L}}$$

  ➜ to achieve 10 Gbps throughput, need a loss rate of L = $2 \cdot 10^{-10}$ – *a very small loss rate!*
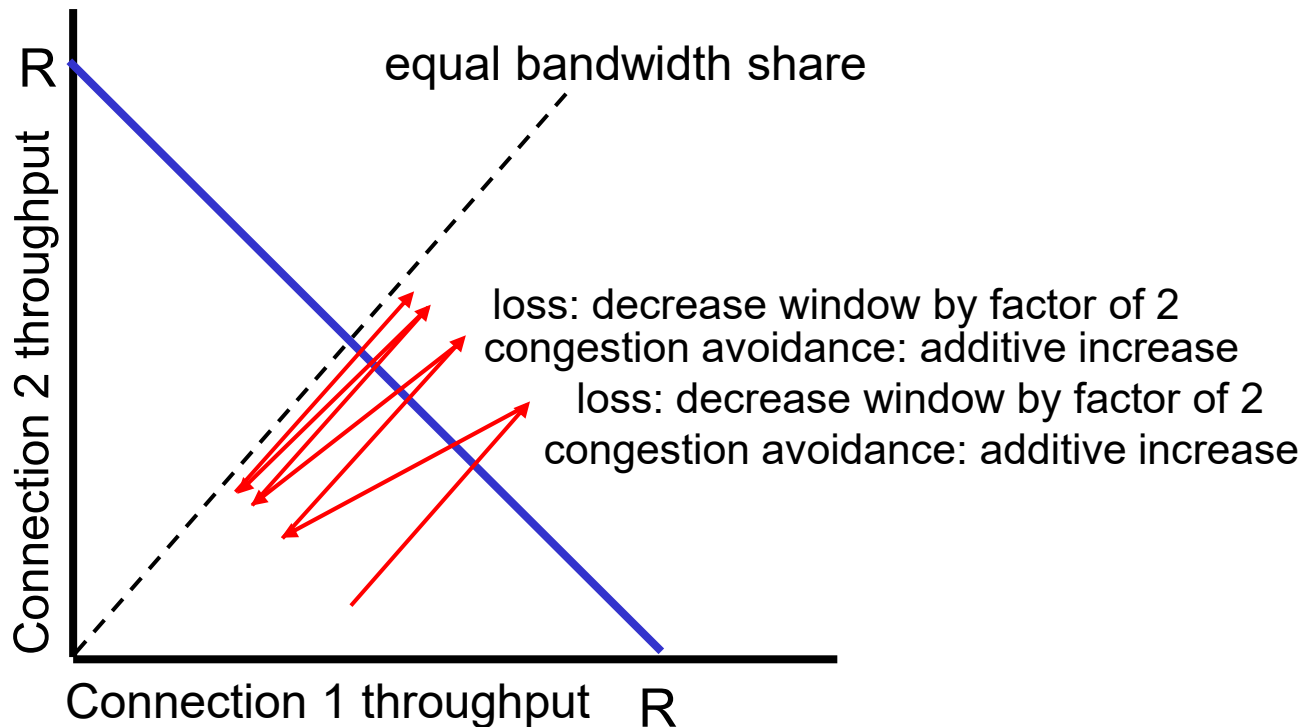- new versions of TCP for high-speed

TCP Fairness

fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K

TCP connection 1

TCP connection 2

bottleneck router capacity R
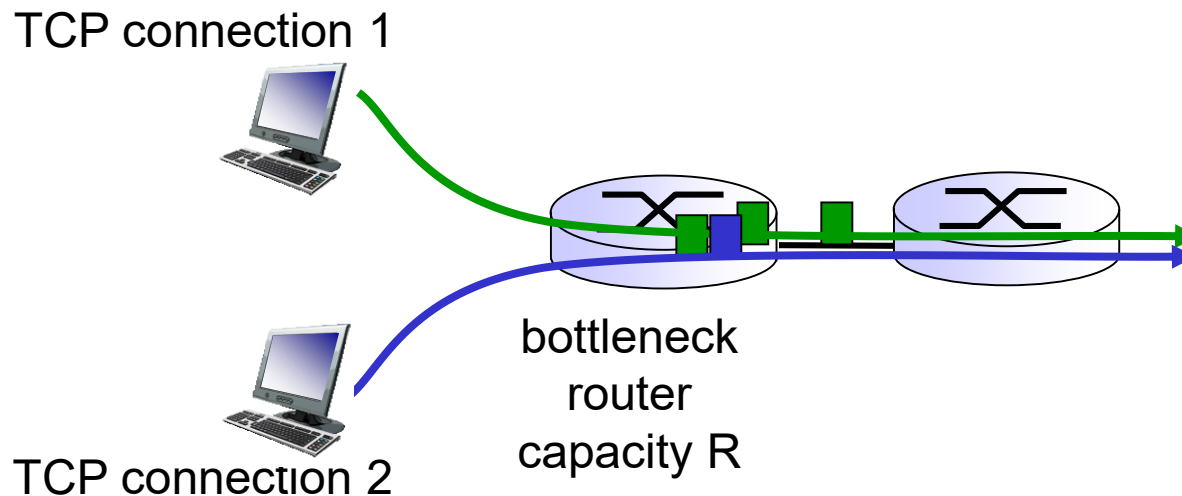
Why is TCP fair?

two competing sessions:
- additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally

equal bandwidth share

loss: decrease window by factor of 2
congestion avoidance: additive increase
loss: decrease window by factor of 2
congestion avoidance: additive increase

Connection 2 throughput

Connection 1 throughput   R

R

Fairness (more)

Fairness and UDP

- multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- instead use UDP:
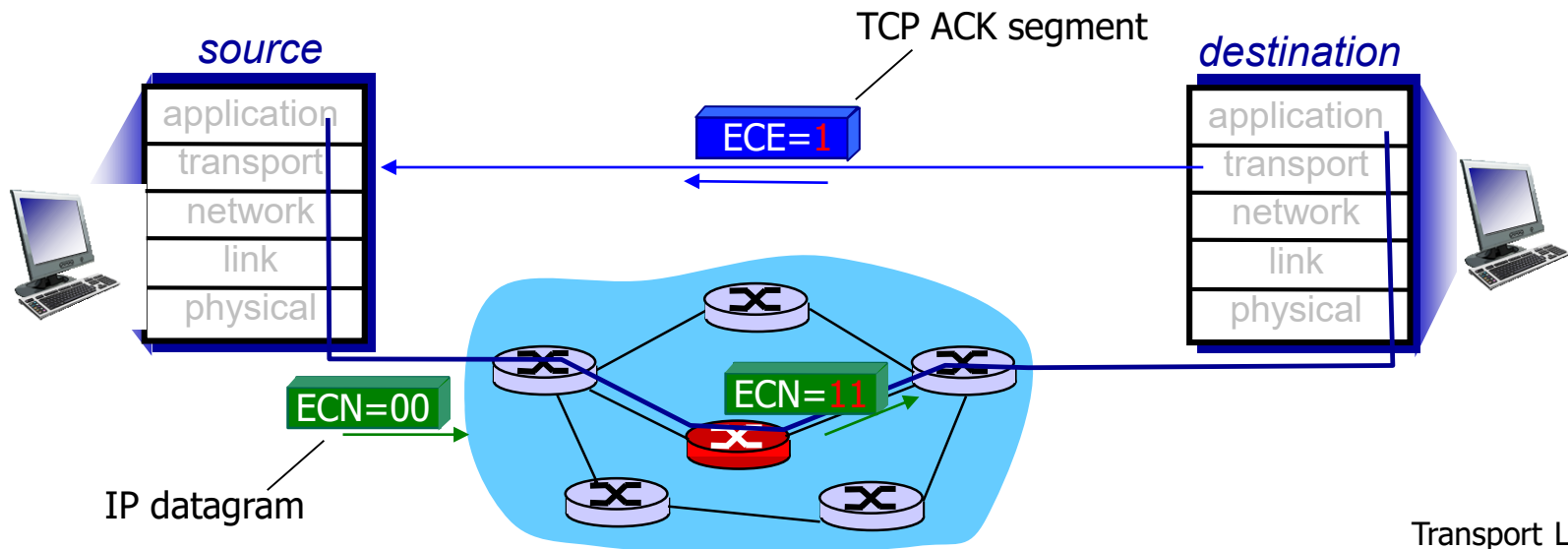 - send audio/video at constant rate, tolerate packet loss

Fairness, parallel TCP connections

- application can open multiple parallel connections between two hosts
- web browsers do this
- e.g., link of rate R with 9 existing connections:
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2

Explicit Congestion Notification (ECN)

network-assisted congestion control:

- two bits in IP header (ToS field) marked *by network router* to indicate congestion
- congestion indication carried to receiving host
- receiver (seeing congestion indication in IP datagram) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion

TCP ACK segment

source

| application |
| transport |
| network |
| link |
| physical |

ECE=1

destination

| application |
| transport |
| network |
| link |
| physical |

ECN=00

ECN=11

IP datagram

Chapter 3: summary

- **principles behind transport layer services:**
  - multiplexing, demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- **instantiation, implementation in the Internet**
  - UDP
  - TCP

next:

- **leaving the network "edge" (application, transport layers)**
- **into the network "core"**
- **two network layer chapters:**
  - data plane
 - control plane

Chapter 4: outline

Chapter 4: network layer

chapter goals:

- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - generalized forwarding
- instantiation, implementation in the Internet

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it
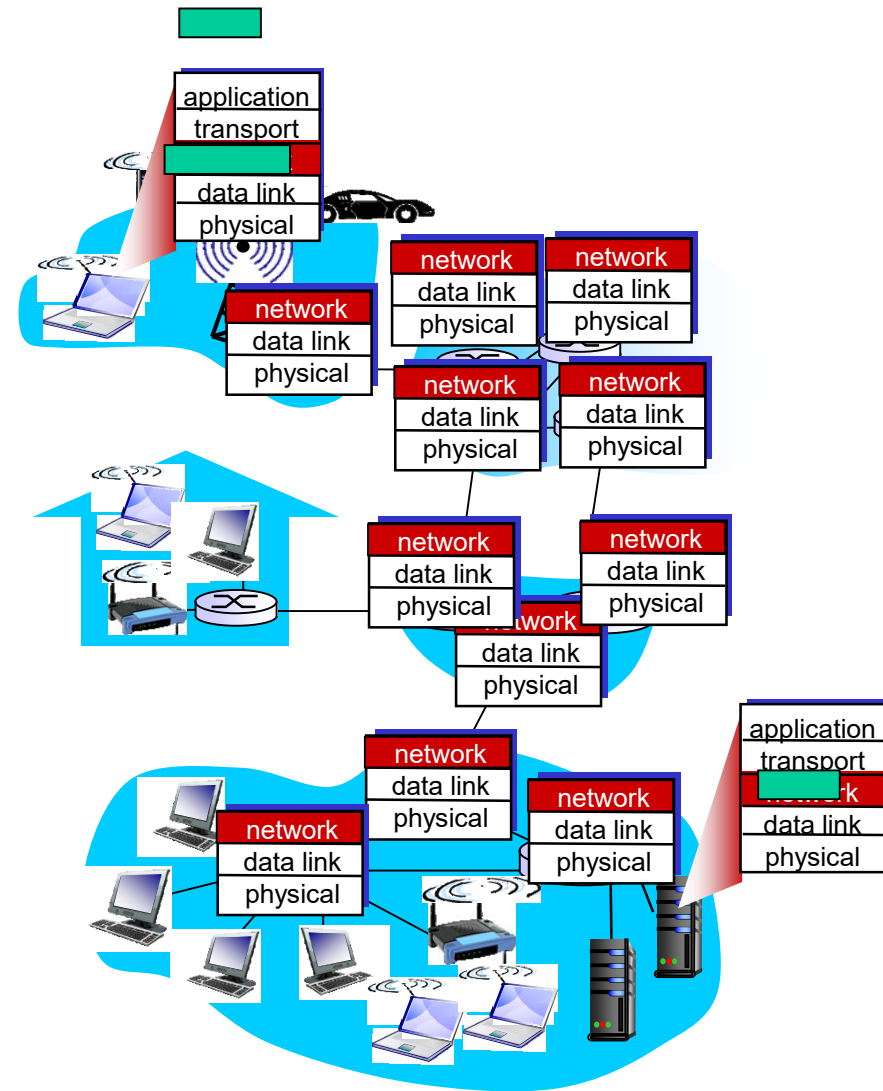
Two key network-layer functions

network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
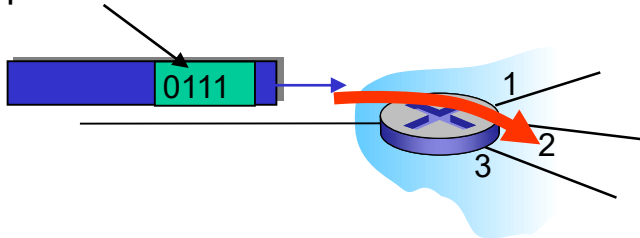 - *routing algorithms*

analogy: taking a trip

- *forwarding:* process of getting through single interchange

- *routing:* process of planning trip from source to destination

Network layer: data plane, control plane

Data plane

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

values in arriving packet header

Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane

Routing Algorithm

Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

control plane

data plane

values in arriving packet header

0111

1
3   2

Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)